**Maven Commands**

1. **mvn clean**: Cleans the project by removing the target directory where the compiled files are stored. This command cleans the Maven project by deleting the target directory:
2. **mvn compile**: Compiles the source code of the project.

**mvn compiler:compile** This command compiles the Java source classes of the Maven project

**mvn compiler:testCompile** This command compiles the test classes of the Maven project:

1. **mvn test**: Runs the unit tests using a suitable testing framework.
2. **mvn package**: Packages the compiled code into a distributable format, such as a JAR or WAR file.
3. **mvn install**: Installs the package into the local repository, making it available for other projects on your local machine. This command builds the Maven project and installs the project files (JAR, WAR, pom.xml, etc.) to the local repository:
4. **mvn deploy**: Deploys the package to a remote repository for sharing with other developers. This command deploys the artifact to the remote repository:
5. **mvn site**: Generates project documentation.
6. **mvn validate** This command validates the Maven project to ensure that everything is correct and all the necessary information is available:
7. **mvn verify :** This command builds the project, runs all the test cases and run any checks on the results of the integration tests to ensure quality criteria are met
8. **mvn -help :** This command-line option prints the Maven usage and all the available options
9. **mvn clean install** : [This will clean the project and then build it from scratch](https://stackoverflow.com/questions/35409788/how-to-clean-or-clean-build-my-maven-project-in-intellij-idea)

**Maven Lifecycle Stages**

Maven has three built-in lifecycles: **default**, **clean**, and **site**. Each lifecycle consists of a sequence of phases.

**Default Lifecycle**

The default lifecycle handles project deployment and consists of the following phases:

1. **validate**: Validates the project is correct and all necessary information is available.
2. **compile**: Compiles the source code.
3. **test**: Runs unit tests.
4. **package**: Packages the compiled code into a distributable format.
5. **verify**: Runs checks on the results of integration tests to ensure quality criteria are met.
6. **install**: Installs the package into the local repository.
7. [**deploy**: Copies the final package to the remote repository for sharing1](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)[2](https://www.baeldung.com/maven-goals-phases).

**Clean Lifecycle**

The clean lifecycle handles project cleaning and consists of the following phases:

1. **pre-clean**: Executes custom tasks before cleaning.
2. **clean**: Removes files generated by the previous build.
3. [**post-clean**: Executes custom tasks after cleaning1](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)[2](https://www.baeldung.com/maven-goals-phases).

**Site Lifecycle**

The site lifecycle handles the creation of the project’s site documentation and consists of the following phases:

1. **pre-site**: Executes tasks needed before generating site documentation.
2. **site**: Generates the project’s site documentation.
3. **post-site**: Executes tasks needed after generating site documentation.
4. [**site-deploy**: Deploys the generated site documentation to a web server1](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)[2](https://www.baeldung.com/maven-goals-phases).

**Example Usage**

To run a specific phase, you can use the mvn command followed by the phase name. For example:

mvn clean

mvn compile

mvn package

Running a phase will also execute all preceding phases in the lifecycle. For example, running mvn package will execute validate, compile, test, and then package.